JSON Messaging

Michael VanSickle
Author

@vansimke

Overview

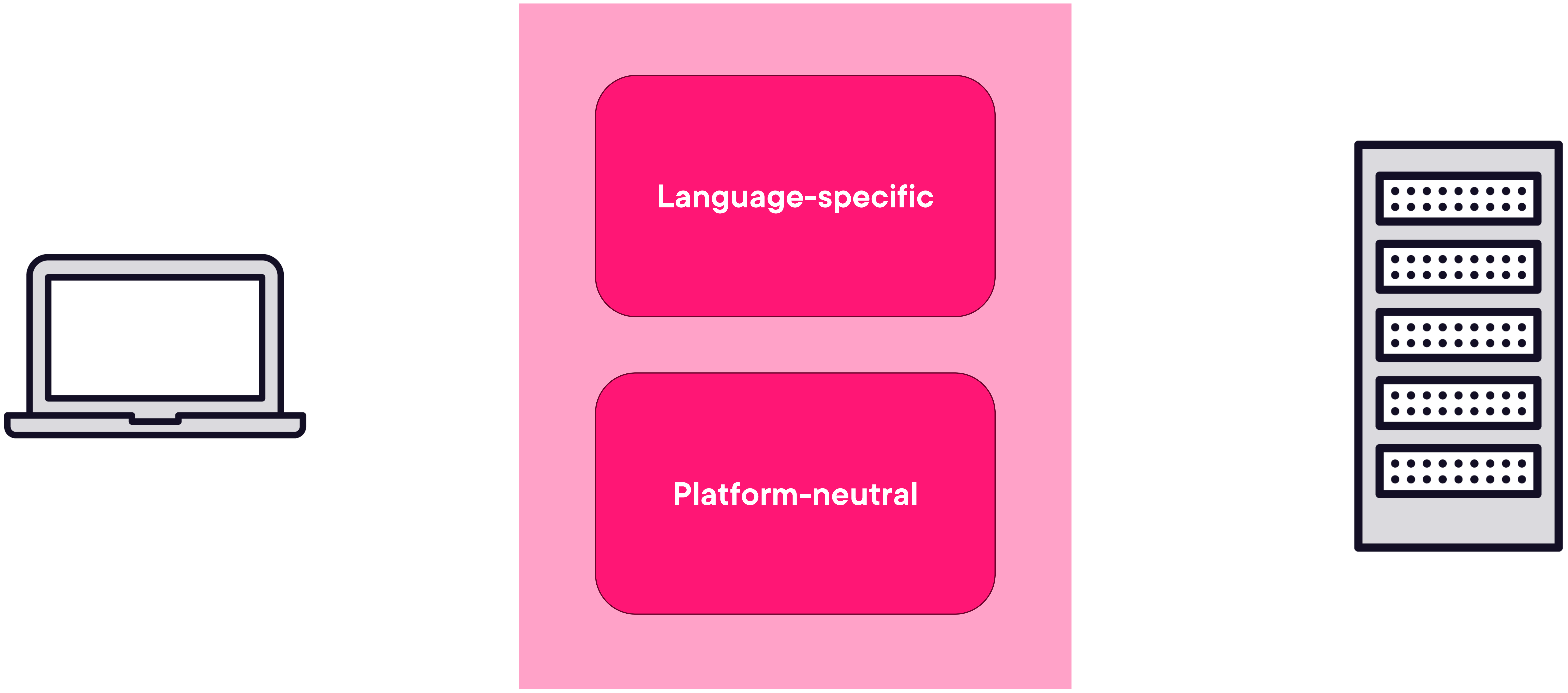
Messaging Strategies

Review of JSON format (quick!)

Sending JSON Messages

Receiving JSON Messages

new customer
customers


Language-specific
Platform-neutral

Messaging Strategies

Language-specific	vs	Platform-neutral
fast		slower
efficient		potentially less efficient
easy to implement		added complexity
reusable message code		reusable message formats
platform lock-in		platform freedom

Platform-neutral Formats

JSON Messaging

gRPC Messaging

```
{
    “id”:           1,
    “firstName”:    “John”,
    “lastName”:     “Smith”,
    “address”:
        “123 Main St, Anytown, USA”
}
```

◂JSON messages enclosed in braces

◂field names are part of message

◂possible data types
- numbers
- strings
- Boolean
- arrays
- objects
- null

Sending JSON Messages

```
import “encoding/json”
import “bytes”

type Customer struct {
    ID          int
    FirstName   string
    LastName    string
    Address     string
}

func convertToJSON(c Customer) ([]byte, error) {
    data, err := json.Marshal(c)
    return data, err
}
```

Sending JSON Messages

```
import “encoding/json”
import “bytes”

type Customer struct {
    ID            int
    FirstName     string
    LastName      string
    Address       string
}

func convertToJSON(c Customer) ([]byte, error) {
    data, err := json.Marshal(c)
    return data, err
}
```

Sending JSON Messages

```
import “encoding/json”
import “bytes”

type Customer struct {
    ID           int
    FirstName    string
    LastName     string
    Address      string
}

func convertToJSON(c Customer) ([]byte, error) {
    data, err := json.Marshal(c)
    return data, err
}
```

Sending JSON Messages

```
import “encoding/json”
import “bytes”

type Customer struct {
    ID          int             `json=“id”`
    FirstName   string
    LastName    string
    Address     string
}

func convertToJSON(c Customer) ([]byte, error) {
    data, err := json.Marshal(c)
    return data, err
}
```

Sending JSON Messages

```
import “encoding/json”
import “bytes”

type Customer struct {
    ID          int          `json=“id”`
    FirstName   string       `json=“firstName”`
    LastName    string       `json=“lastName”`
    Address     string       `json=“address”`
}

func convertToJSON(c Customer) ([]byte, error) {
    data, err := json.Marshal(c)
    return data, err
}
```

Sending JSON Messages

```
import “encoding/json”
import “bytes”

type Customer struct {
    ID          int         `json=“id”`
    FirstName   string      `json=“firstName”`
    LastName    string      `json=“lastName”`
    Address     string      `json=“address”`
}

func convertToJSON(c Customer) ([]byte, error) {
    var b bytes.Buffer
    enc := json.NewEncoder(b)
    err := enc.Encode(c)
    return b.Bytes(), err
}
```

Receiving JSON Messages

```
import “encoding/json”
import “bytes”

type Customer struct {
    ID          int          `json=“id”`
    FirstName   string       `json=“firstName”`
    LastName    string       `json=“lastName”`
    Address     string       `json=“address”`
}

func convertFromJSON(data []byte) (Customer, error) {


}
```

Receiving JSON Messages

```
import “encoding/json”
import “bytes”

type Customer struct {
    ID            int          `json=“id”`
    FirstName     string       `json=“firstName”`
    LastName      string       `json=“lastName”`
    Address       string       `json=“address”`
}

func convertFromJSON(data []byte) (Customer, error) {

    err := json.Unmarshal(data)
    return c, err
}
```

Receiving JSON Messages

```
import “encoding/json”
import “bytes”

type Customer struct {
    ID          int         `json=“id”`
    FirstName   string      `json=“firstName”`
    LastName    string      `json=“lastName”`
    Address     string      `json=“address”`
}

func convertFromJSON(data []byte) (Customer, error) {
    var c Customer
    err := json.Unmarshal(data, &c)
    return c, err
}
```

Receiving JSON Messages

```
import “encoding/json”
import “bytes”

type Customer struct {
    ID          int          `json=“id”`
    FirstName   string       `json=“firstName”`
    LastName    string       `json=“lastName”`
    Address     string       `json=“address”`
}

func convertFromJSON(data []byte) (Customer, error) {

    dec := json.NewDecoder( )



}
```

Receiving JSON Messages

```
import “encoding/json”
import “bytes”

type Customer struct {
    ID           int             `json=“id”`
    FirstName    string          `json=“firstName”`
    LastName     string          `json=“lastName”`
    Address      string          `json=“address”`
}

func convertFromJSON(data []byte) (Customer, error) {
    b := bytes.NewBuffer(data)        // must be an io.Reader
    dec := json.NewDecoder(b)



}
```

Receiving JSON Messages

```
import “encoding/json”
import “bytes”

type Customer struct {
    ID          int          `json=“id”`
    FirstName   string       `json=“firstName”`
    LastName    string       `json=“lastName”`
    Address     string       `json=“address”`
}

func convertFromJSON(data []byte) (Customer, error) {
    b := bytes.NewBuffer(data)       // must be an io.Reader
    dec := json.NewDecoder(b)

    err := dec.Decode(&c)
    return c, err
}
```

Receiving JSON Messages

```
import “encoding/json”
import “bytes”

type Customer struct {
    ID          int         `json=“id”`
    FirstName   string      `json=“firstName”`
    LastName    string      `json=“lastName”`
    Address     string      `json=“address”`
}

func convertFromJSON(data []byte) (Customer, error) {
    b := bytes.NewBuffer(data)          // must be an io.Reader
    dec := json.NewDecoder(b)
    var c Customer                      // could use map[string]any too
    err := dec.Decode(&c)
    return c, err
}
```

Summary

Review of JSON format (quick!)

Sending JSON Messages

Receiving JSON Messages